Predicting Bitcoin Price Movement By Machine Learning Models

Irene YiJu Su, Khanh Nguyen, Xinmei Luan

Introduction

Problem Statement

Bitcoin, while sharing some similarities with the traditional stock market, is significantly more volatile due to its decentralized and unregulated nature (Fig. 1. Comparison between BTC Bitcoin price and S&P500, NASDAQ, and DJ Dow Jones index). Speculating on Bitcoin can be hugely rewarding, but it also involves high risk. Therefore, it is critical for Bitcoin investors to make informed decisions about when to enter the market (buy) and when to exit the market (sell).

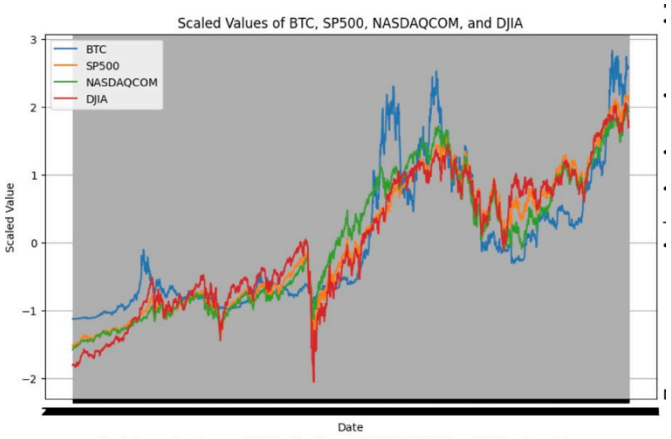


Fig. 1. Comparison between BTC Bitcoin price and S&P500, NASDAQ, and DJ Dow Jones index

Impact of Solving the Problem

By developing a robust methodology for predicting Bitcoin movements, we can potentially reduce the risks associated with Bitcoin trading. Accurate predictions can help investors make better-informed decisions, ultimately leading to more stable investment outcomes and potentially attracting more participants to the Bitcoin market.

Motivation

Our motivation to work on this project stems from the high volatility and associated risks of the Bitcoin market. The need for reliable predictive models is paramount for investors looking to navigate this complex and fast-paced environment. By addressing this need, we aim to contribute to the financial community by providing tools that can enhance trading strategies and risk management practices.

Project Summary

1. Dataset 1: BTC data (plus self-implemented technical indices), call it df0
2. Dataset 2: Dataset 1 combined with 19 FRED indices, call it df1
3. Dataset 3: Dataset 2 plus additional 22 FRED indices, call it df2

We employed various supervised models for BTC movement prediction and an unsupervised model for BTC movement trend clustering. The supervised models included RandomForest, DecisionTree, KNN, SVM, NaiveBayes, and LogisticRegression. The reason we chose these models is due to referenced from our related work #1 Using Supervised Machine Learning Models to Predict Equity Price Movements. The train_test_split works on the **first 80%** of the full set, so the model *fits* on the *train set of train_test_split* and *predicts* on the *test set of train_test_split.* We do *forward prediction* on the **last 20%**, around 500 points of the full set. Three methodologies are implemented for forward prediction to compare their performance.

The unsupervised approach involved k-clustering techniques to identify underlying trends in BTC movements.

Novel Contributions

Our novel contributions compared to related projects include:

1. The implementation of day-by-day forward prediction methodologies, revealing insights into the temporal robustness of each model.
2. An exploration of the impact of macroeconomic indices (FRED) on BTC movement predictions.
3. The merge of feature importance from RandomForest and MLPClassifier, along with systematic correlation analysis and Variance Inflation Factor (VIF) to create a robust and minimal feature set. This approach reduces redundancy and eliminates multicollinearity, enhancing model stability and interpretability.
4. The segmentation of time series data into smaller windows before clustering helps detect small periods of price pattern. This analysis gives more insights into the BTC price movement over short time periods.

Main Findings

1. RandomForest outperforms DecisionTree, KNN, SVM, NaiveBayes, and LogisticRegression in terms of accuracy when fitting the model on the train set and predicting on the test set.
2. Surprisingly, KNN and SVM achieve similar or even better accuracy to RandomForest in forward prediction.
3. Both NaiveBayes and LogisticRegression perform poorly in both model fitting/predict (train-test split) and forward prediction.
4. Despite being a tree-based algorithm like RandomForest, DecisionTree does not perform well.
5. Using shorter periods (e.g., 100 days) for model fitting before predicting the next day's movement yields the best BTC movement predictions.
6. Overall, day-by-day forward prediction performs worse than train-test split predictions, except for KNN and SVM, which perform better in forward predictions.
7. The best accuracy achieved in forward predictions is 0.648 with KNN. This demonstrates BTC movement prediction is not an easy task even though we predict day by day.
8. The integration of FRED indices with BTC data assists model fitting and prediction (train-test split) compared to BTC data alone.
9. K-Shape clustering effectively segmented the BTC time series data into five distinct groups with unique patterns and trends.
10. Sensitivity analysis highlighted that larger window sizes improved clustering performance, with a 50-day window size yielding the best results.
11. PCA facilitated dimensionality reduction and feature extraction, with the first two principal components capturing the most significant variance.

Related Work

1. Using Supervised Machine Learning Models to Predict Equity Price Movements by Jimisi Rajasombat. The study applied various supervised algorithms (RandomForest, KNN, DecisionTree, SVM, NaiveBayes, LogisticRegression, GradientBoosting) to predict stock price movements for up to 60 trading days. It concluded that short-term predictions are no better than stochastic predictions (<50% accuracy) while long-term predictions (40-60 days) could achieve 70% accuracy, highlighting the difficulty of short-term trading strategies. Given that BTC is harder to predict than stocks due to its severe fluctuations, we challenged this conclusion by focusing on the even more challenging task of predicting BTC movements 5 days later using the same algorithms, except GradientBoosting. Rajasombat's work demonstrated the best model is RandomForest for all (1-60) days prediction and his best prediction accuracy for 5-day is 55%. Our work demonstrates RandomForest, KNN, and SVM can achieve >62% accuracy with KNN achieving 65% accuracy.
2. Cryptocurrency Prices and News Sentiment: Which Exerts Influence on the Other?. This paper investigates the relationship between cryptocurrency prices and news sentiment for sentiment analysis. The study reveals a positive correlation between news sentiments and cryptocurrency returns. Interestingly, Korean news sentiment is found to predict global market prices up to a week, as evidenced by the Korean Won surpassing the US Dollar in Bitcoin trading. The sentiment analysis is our plan. However, after long times of searching, we did not find good ways to extract twitter posts and get news content for free. But some indices like funding rate, fear and greed which could be part of sentiment analysis. Due to the length of these two indices being shorter than our dataset, we decided to move this part of analysis to our future research.
3. Clustering and Modelling of the Top 30 Cryptocurrency Prices Using Dynamic Time Warping and Machine Learning Methods byTomáš Šťastný. This is the study we referenced for Unsupervised learning. This paper provides a comprehensive analysis of the crypto prices using a set of advanced time series analysis methods. It employs dynamic time warping (DTW) to calculate optimal alignments between crypto price movements and uses k-Shape clustering to group similar cryptocurrencies based on those movements. However, we will focus only on BTC price movements and run clustering on different time windows to find groups with similar movements. Knowing which pattern group a data point might belong to would be helpful when we classify the price movements.

Supervised learning

Data Sources

Source	Location/Resource	Format Returned/Used	Important Variables	Number of Records	Time Period Covered

BTC Data	Yahoo Finance	CSV	Open, Close,High, Low, Volume, Technical index macd_hist, macd_signal, rsi, macd	2684	2017-01-01 to 2024-06-01
FRED Index	FRED website (via fredapi)	JSON (converted to pandas DataFrame)	UNRATE, WM2NS, DJIA, NASDAQCOM,	1865	2017-01-01 to 2024-06-01

Initial Preprocessing:

1. Handling Missing Data: Forward fill (ffill) and backward fill (bfill) were used to handle null values in the merged FRED data due to varying frequencies (daily, weekly, monthly, quarterly).
2. Given our dataset spanning more than 7 years, we aimed to identify outliers using UMAP and LocalOutlierFactor. Our initial attempt identified the top 20 outliers ([1517, 1515, 1516, 1518, 1519, 1520, 1521, 1522, 1178, 1488, 1487, 1489, 1490, 1491, 1493, 1492, 2427, 2428, 1180, 1175]). Among these, we observed two consecutive periods with more than 7 points each (1515-1522 and 1487-1493), suggesting these are not true outliers but reflect significant events during those periods. To maintain dataset's integrity and avoid removing potentially valuable information, we decided not to exclude these points

Feature Engineering

Implement Technical Indices from BTC Data (output: df0):
- RSI (Relative Strength Index)
- Moving Average: SMA (Simple Moving Average), EMA (Exponential Moving Average),
- Moving Average Convergence Divergence: MACD, MACD Signal, MACD Histogram

Many Feature Engineering works are shared with the unsupervised pipeline.

Final Feature Set: see appendix

Methods Description

1. Target setting
   To overcome data distribution imbalance, we classified movement with a **Threshold** of Close (Closing price) * 0.0275.

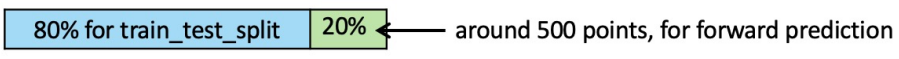
$$\text{target} = \begin{cases} 1 & \text{if } \text{Close}_{\text{5-day-later}} > \text{Close}_{\text{current}} \times 1.0275 \text{ (Up)} \\ -1 & \text{if } \text{Close}_{\text{5-day-later}} < \text{Close}_{\text{current}} \times 0.9725 \text{ (Down)} \\ 0 & \text{if } \text{Close}_{\text{current}} \times 0.9725 \leq \text{Close}_{\text{5-day-later}} \leq \text{Close}_{\text{current}} \times 1.0275 \text{ (Equal} \end{cases}$$

target	total
1	994
0	955
-1	735

   This setting reflects BTC's volatile movement but also takes care of data imbalance

2. Datasets: We want to investigate whether FRED index integration assists prediction.
   df0: BTC data with technical index.
   df1: df0 plus 19 FRED index.
   df2: df1 plus 22 FRED index (i.e., df0+ 41 FRED index).

3. Methodology:
   Separate the full set into 2 parts: First 80% and last 20% (around 500 points)



   In first 80% we do following to get model performance baseline
   - GridSearch on the train set to find best model
   - Best model fit on the train set
   - Best model predict on the test set

   Then we do forward prediction on the last 20% with three methodologies.

   Methodology #1: Long-day fit then predict, day by day

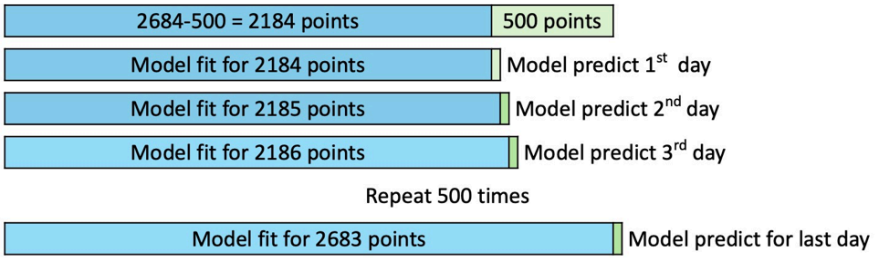


Figure 2: Methodology #1

Methodology #2: Short-day fit then predict, day by day

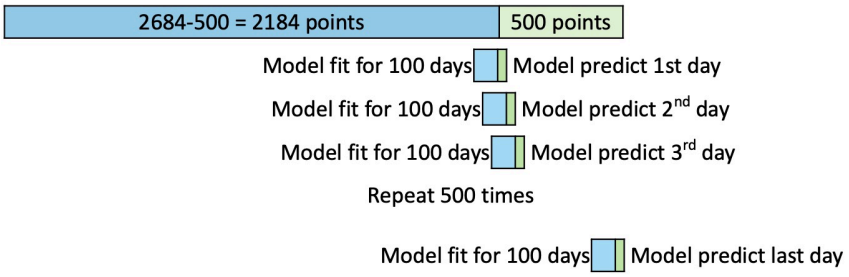


Figure 3: Methodology #2

Methodology #3: Split the last 20%, 500 points, into 5 groups. Long-day GridSearch and fit then predict, group by group

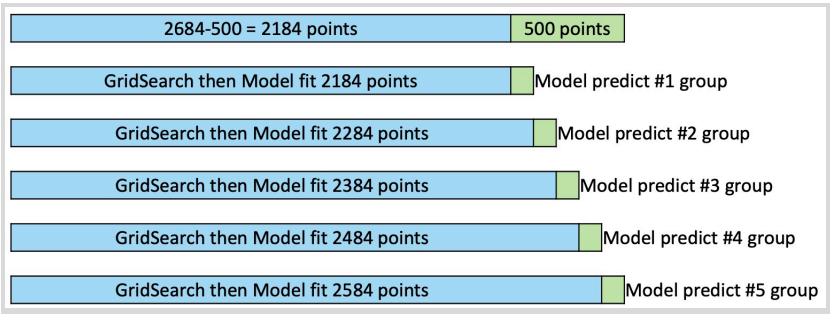


Figure 4: Methodology #3

Supervised evaluation

Overall Results

Evaluation Metrics

We chose the following evaluation metrics for assessing the performance of our models in the context of a three-label classification task (labels: -1, 0, 1):

1. Accuracy: Measures the overall correctness of the model's predictions. It is straightforward by indicating the proportion of correct predictions out of the total predictions. Accuracy is an evaluation metric we applied in all GridSearch, model predict for train_test_split test set and forward prediction.
2. F1 Score: Provides a balance between precision and recall, especially useful in cases where the class distribution is imbalanced. It gives a single metric that considers both false positives and false negatives across all three classes (-1, 0, 1). F1_score is an evaluation metric we applied in all model predictions for train_test_split test set and forward prediction.
3. Recall: Recall is the ratio of correctly predicted positive observations to the all observations in actual class. High Recall indicates the model is capturing most of the actual positive cases (low false negatives). Since our work is 3-type classification, we apply weighted-average to calculate the recall for each class independently and then take the average of these recalls, weighted by the number of true instances for each class. Recall is an evaluation metric we applied in all model predictions for the train_test_split test set.
4. Precision: Precision is the ratio of correctly predicted instances of that class to the total instances predicted as that class. High Precision indicates that the model is predicting positive cases with high accuracy (low false positives). Since our work is 3-type classification, we apply weighted-average to calculate the precision for each class independently and

then take the average of these precisions, weighted by the number of true instances for each class. Precision is an evaluation metric we applied in all model predictions for the train_test_split test set.

5. Confusion Matrix: Offers a detailed breakdown of true positives, true negatives, false positives, and false negatives for each class (-1, 0, 1), providing insights into the types of errors the model is making and the distribution of correct and incorrect predictions across the three classes. Confusion Matrix is an evaluation metric we applied in all model predictions for train_test_split test set and forward prediction.
6. In addition to the five evaluation metrics mentioned above, we assess model performance using the test set performance of the first 80% of the data and the forward prediction performance of the last 20%.

Summary of Results (only best model for each family output here)

- **CV result and test set prediction result (df0 & df1): Confusion Matrix in appendix-C**

Model Family	Best Model	CV Accuracy (Mean ± Std)	Accuracy	f1_score	Recall	Precision
Tree-Based Models	RandomForest	0.610 ± 0.014	0.689	0.689	0.689	0.689
Instance-Based Models	KNN	0.539 ± 0.037	0.606	0.608	0.606	0.609
Support Vector Machines	SVM	0.440 ± 0.060	0.609	0.610	0.609	0.613
Bayesian Models	Naive Bayes	0.397 ± 0.001	0.547	0.548	0.547	0.550
Linear Models	Logistic Regression	0.409 ± 0.008	0.405	0.405	0.405	0.418

Table 5: Cross validation result for first dataset,df0.

Model Family	Best Model	CV Accuracy (Mean ± Std)	Accuracy	f1_score	Recall	Precision
Tree-Based Models	RandomForest	0.659 ± 0.017	0.714	0.713	0.714	0.713
Instance-Based Models	KNN	0.585 ± 0.041	0.696	0.695	0.696	0.695
Support Vector Machines	SVM	0.484 ± 0.085	0.604	0.605	0.604	0.613
Bayesian Models	Naive Bayes	0.403 ± 0.002	0.668	0.665	0.668	0.671
Linear Models	Logistic Regression	0.445 ± 0.008	0.437	0.400	0.437	0.464

Table 6: Cross validation result for the second dataset,df1.

- **Forward Prediction result: methodology #1 long day fit then predict vs #2 short-day fit then predict (on df0 & df1)**

df0: BTC+technical index		(500 days prediction)		accuracy rate on each classification		
Model Family	Best Model	accuracy	f1_score	-1	0	1
Tree-Based Models	RandomForest	0.58	0.578	0.500	0.669	0.480
Instance-Based	KNN	0.596	0.589	0.479	0.728	0.447
Support Vector Machines	SVM	0.542	0.545	0.574	0.563	0.487
Bayesian Models	NaiveBayes	0.424	0.393	0.372	0.646	0.086
Linear Models	LogisticRegression	0.462	0.416	0.085	0.740	0.230
df1: df0+19 FRED index		(500 days prediction)		accuracy rate on each classification		
Model Family	Best Model	accuracy	f1_score	-1	0	1
Tree-Based Models	RandomForest	0.644	0.641	0.462	0.719	0.630
Instance-Based	KNN	0.64	0.638	0.527	0.715	0.584
Support Vector Machines	SVM	0.638	0.636	0.505	0.708	0.604
Bayesian Models	NaiveBayes	0.508	0.364	0.065	0.980	0.000
Linear Models	LogisticRegression	0.494	0.430	0.333	0.806	0.078

df0: BTC+technical index		(500 days		accuracy rate on each classification		
Model Family	Best Model	accuracy	f1_score	-1	0	1
Tree-Based Models	RandomForest	0.638	0.634	0.436	0.720	0.625
Instance-Based	KNN	0.62	0.614	0.415	0.736	0.553
Support Vector Machines	SVM	0.632	0.624	0.489	0.776	0.480
Bayesian Models	NaiveBayes	0.412	0.416	0.564	0.441	0.270
Linear Models	LogisticRegression	0.448	0.443	0.394	0.598	0.230
df1: df0+19 FRED index		(500 days		accuracy rate on each classification		
Model Family	Best Model	accuracy	f1_score	-1	0	1
Tree-Based Models	RandomForest	0.642	0.639	0.505	0.723	0.591
Instance-Based	KNN	0.648	0.647	0.538	0.719	0.597
Support Vector Machines	SVM	0.634	0.620	0.430	0.822	0.448
Bayesian Models	NaiveBayes	0.496	0.502	0.581	0.502	0.435
Linear Models	LogisticRegression	0.564	0.555	0.387	0.719	0.416

Table 7: Forward prediction result, evaluated using methodology #1.

- **Methodology #3 using two methods, which are train_test_split and forward testing.**

df2- 58 features	Model	Best Parameters	Accuracy
0	RandomForest	{'max_depth': 20, 'min_samples_split': 2, 'n_e=50}	0.739292
1	DecisionTree	{'max_depth': None, 'min_samples_split': 2}	0.64432
2	LogisticRegression	{'C': 10, 'max_iter': 100, 'multi_class': 'ovr'}	0.50838
3	KNN	{'n_neighbors': 3}	0.692737

df_shorter 12 features	Model	Best Parameters	Accuracy
0	RandomForest	{'max_depth': None, 'min_samples_split': 2, 'n_e=150}	0.726257
1	DecisionTree	{'max_depth': None, 'min_samples_split': 2}	0.612663
2	LogisticRegression	{'C': 10, 'max_iter': 100, 'multi_class': 'ovr'}	0.459963
3	KNN	{'n_neighbors': 3}	0.689013

Table 8: train_test_split comparison.

df2, 58 features ,5 groups*100 days	accuracy	precision_-1	recall_-1	f1_score_-1	precision_0	recall_0	f1_score_0	precision_1	recall_1	f1_score_1
RandomForest	39.4%	21.7%	33.0%	26.2%	48.4%	54.3%	51.2%	38.9%	18.4%	25.0%
DecisionTree	28.6%	20.7%	52.1%	29.6%	45.5%	29.5%	35.8%	19.4%	12.5%	15.2%
LogisticRegression	50.4%	31.3%	16.0%	21.1%	52.4%	93.3%	67.1%	0.0%	0.0%	0.0%
KNN	39.0%	27.9%	18.1%	21.9%	52.2%	36.6%	43.1%	32.6%	55.9%	41.2%
df-shorter,12 features,5 groups*100 days	accuracy	precision_-1	recall_-1	f1_score_-1	precision_0	recall_0	f1_score_0	precision_1	recall_1	f1_score_1
RandomForest	39.2%	26.6%	48.9%	34.5%	53.2%	52.4%	52.8%	22.1%	11.2%	14.8%
DecisionTree	37.2%	22.6%	51.1%	31.4%	51.9%	44.1%	47.7%	36.1%	17.1%	23.2%
LogisticRegression	50.2%	29.8%	18.1%	22.5%	52.9%	90.9%	66.9%	50.0%	2.0%	3.8%
KNN	32.4%	16.9%	31.9%	22.1%	53.0%	34.3%	41.6%	28.5%	29.6%	29.0%
df2, 58 features ,5 groups*10 days	accuracy	precision_-1	recall_-1	f1_score_-1	precision_0	recall_0	f1_score_0	precision_1	recall_1	f1_score_1
RandomForest	56.0%	54.2%	72.2%	61.9%	53.3%	44.4%	48.5%	63.6%	50.0%	56.0%
DecisionTree	28.0%	27.8%	27.8%	27.8%	17.6%	16.7%	17.1%	40.0%	42.9%	41.4%
LogisticRegression	36.0%	0.0%	0.0%	0.0%	36.0%	100.0%	52.9%	0.0%	0.0%	0.0%
KNN	50.0%	52.9%	50.0%	51.4%	64.3%	50.0%	56.3%	36.8%	50.0%	42.4%
df-shorter,12 features,5 groups*10 days	accuracy	precision_-1	recall_-1	f1_score_-1	precision_0	recall_0	f1_score_0	precision_1	recall_1	f1_score_1
RandomForest	56.0%	48.1%	72.2%	57.8%	64.3%	50.0%	56.3%	66.7%	42.9%	52%
DecisionTree	34.0%	27.3%	33.3%	30.0%	50.0%	44.4%	47.1%	25.0%	21.4%	23%
LogisticRegression	40.0%	0.0%	0.0%	0.0%	37.0%	94.4%	53.1%	75.0%	21.4%	33%
KNN	38.0%	52.9%	50.0%	51.4%	0.0%	0.0%	0.0%	31.3%	71.4%	43%

Table 9: Forward testing method with 100 days and 10 days forecast.

Explanation of Results

Overall

For df0 & df1:

1. df1 (FRED index integrated with BTC + technical index), performs better than df0 (BTC + technical index only) across all models. It demonstrates FRED index is beneficial to model prediction.
2. Overall, forward prediction accuracy decreases compared to test set prediction, with a drop ranging from 0.05 to 0.1.
3. F1_score, recall, precision all perform the same level as accuracy among all models in both datasets. This reveals the models are likely performing consistently across different metrics. This suggests that the models ares not only good at making correct predictions (accuracy) but also have good balance between precision (the accuracy of positive predictions) and recall (the ability to find all positive instances).
4. Confusion matrix for the test set (in train_test_split) shows target 0's accuracy rate is worst. But in forward prediction, target 0's oppositely gets best accuracy. This holds true for both datasets and for both Methodology#1 & Methodology#2.

For df2:

1. 12 features performing are slightly worse in train_test_spit method than 58 features. However, in the forward testing part, they are performing almost the same apart from KNN.
2. RandomForest, KNN, and DecisionTree show moderate to high accuracy on the test set other than LogisticRegression, ,ranging from 0.644 to 0.739. All models perform poorly in forward testing with the same test size. However, when test size shortened to 10 days from 100 days, forward testing results improved obviously. Test size shorten to 1 days result reflect in methodology #2.
3. The 12 features are close,high,open,low,volume,month,day,MACD,MACD-signal,Relative Strength Index,Consumer Price Index: Total All Items for the United States,Producer Price Index by Industry: Financial Services.

Model Comparison

Tree-Based Models: RandomForest:

- Performance: RandomForest outperforms in both cross-validation (CV) and test set for both datasets. In df1, RandomForest shows an accuracy of 0.714.
- Forward Prediction: However, in forward prediction, RandomForest shows similar accuracy to KNN and SVM. Specifically, in methodology #2 short-day-fit forward prediction, RandomForest is not the best performer; KNN is.
- Possible reasons why RandomForest doesn't consistently outperform KNN and SVM:
    - RandomForest may over-rely on long-term patterns identified in extensive historical data, which might not generalize well to new, unseen data, especially in the short-term. This over-reliance can cause it to underperform in dynamic, short-term contexts where recent changes are more critical.

target	total	use4TrainTest	use4Froward
1	994	842	152
0	955	701	254
−1	735	641	94

    - RandomForest might struggle to adapt quickly if the forward prediction set has a significantly different distribution from the training set. Data points used for train_test_split have a comparable equal distribution of target -1/0/1 (641:701:842) but in the forward prediction set, the distribution is quite imbalanced (94:254:152). This lack of adaptability to new distribution patterns can impact its forward prediction performance.

Instanced-Based: KNN and Support Vector Machines: SVM:

- Performance: KNN and SVM perform moderately in both cross-validation (CV) and the train-test split test set for both datasets.
- Forward Prediction: They perform well in forward prediction, particularly in short-day-fit predictions.
- Possible Reasons:
    - KNN and SVM might be more robust in capturing local patterns and trends within the data, which becomes advantageous in forward prediction scenarios where short-term predictions are required. Their simplicity and flexibility in adapting to new data patterns might contribute to their better performance in short-day-fit forward prediction, where the model needs to quickly adjust to recent data.
    - KNN is a local learning algorithm that can perform well on imbalanced datasets if the majority classes are correctly predicted by nearby neighbors. The imbalanced forward prediction set might benefit KNN, as it relies on local patterns which can still be effective despite the imbalance

Naive Bayes and Linear Models: Logistic Regression (using lbfgs solver):

- Performance: Naive Bayes and Logistic Regression do not perform well in either the train-test split test set or forward prediction. Their predictions are no better than stochastic predictions.
- Possible Reasons: These models may struggle due to their underlying assumptions.
    - Naive Bayes assumes feature independence, which is rarely the case in financial data.
    - Logistic Regression is fundamentally a linear model and, despite using the lbfgs solver, may not capture complex patterns as effectively as other non-linear models.

Overall df2's result using methodology #3

- Performance: perform the same level as those works of df0/df1, and RandomForest performs best, too.
- Forward Prediction: all models perform poorly with the same test size above.
- Possible Reasons:
    - We check different targets by precision/recall (not in the original evaluation metrics list) additionally. It shows all targets have similar precision/recall, thus not unique type's issue.
    - Model fit on random data might not be good at predicting consecutive 100 days. Detailed will be discussed in Failure Analysis Section

Feature Importance Analysis

(Feature Importance Analysis is done on both df1 and df2. Since df2 has more than 50 features, here we put our work of feature importance in df2 only)

Handling Multicollinearity:

1. Apply correlation matrix to better understand the relationships among features.
   Strong Positive Correlations
    - in Top Left Corner: Open, Close, High, Low, sma, ema
    - Macd group: macd, macd_signal
    - Stock related Index Group: SP500, NASDAQCOM, DJIA
    - Macroeconomic index: (CP, RSXFS), (UNRATE, CGBD2534)

Negative Correlations:

- FEDFUNDS and UMCSENT (University of Michigan: Consumer Sentiment): The two features exhibited a negative correlation with most other features.
- UNRATE (Unemployment Rate) and INDPRO (Industrial Production: Total Index): These features also showed negative correlations with many other features.

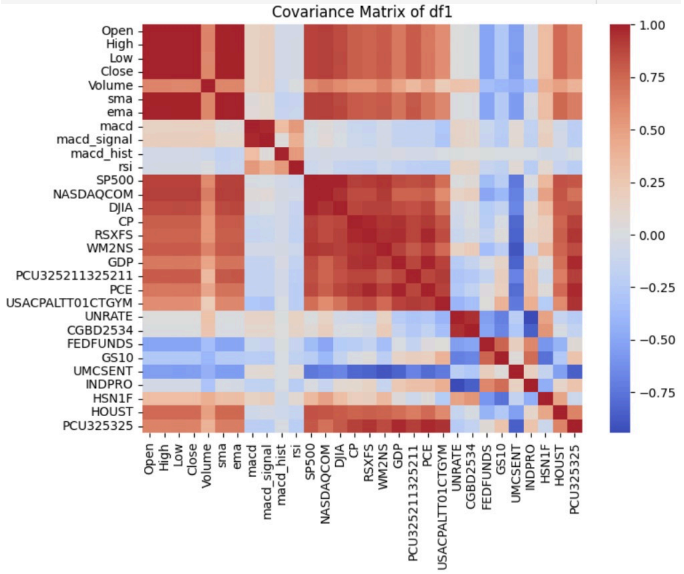


Figure 10: Covariance matrix of the BTC price and 41 FRED indices.

2. Set correlated feature threshold:
   - Set a correlation threshold of 0.95 to identify highly correlated feature pairs.
   - Sorted features by their occurrence in these pairs and visualized the results in a bar chart.
   - CPIAUCSL (consumer price index for all urban consumers) had the highest correlation with other features.
   - This analysis identified the most interrelated features, guiding further steps to address multicollinearity. Below plot, left)
3. VIF (Variance Inflation Factor) Analysis
   - Used the Variance Inflation Factor (VIF) to measure the increase in variance of an estimated regression coefficient due to multicollinearity.
   - Combined VIF results with correlation analysis to identify problematic features.
   - Notable features with high correlations and high VIF values included SMA, EMA, and GDP, indicating significant multicollinearity.
   - Outcome: This combined approach allowed for a more effective identification and addressing of multicollinearity in the dataset.(below plot, right)

Feature Importance Analysis

- From work on df0 and df1, we found RandomForest performs best among 6 models, so we chose RandomForest for feature importance study. From RandomForestClassifier.feature_importance_ we get TOP 20 most important features as baseline (Figure 11)
- We additionally apply MLP classifiers to find feature importance (Figure 12). It shows against the baseline.

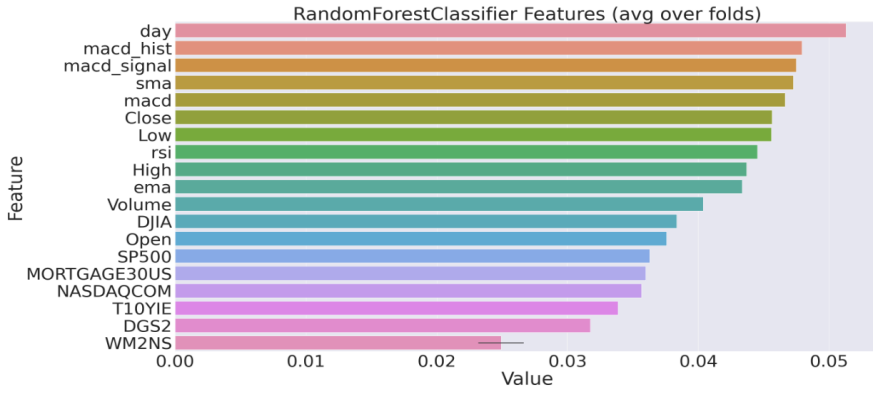


Figure 11: Feature importance from RandomForest classifier

```
Close    0.266 +/- 0.019
DFF      0.203 +/- 0.015
CORESTICKM159SFRBATL0.200 +/- 0.019
DGS2     0.194 +/- 0.017
CPALTT01USM657N0.181 +/- 0.020
STICKCPIM157SFRBATL0.171 +/- 0.016
GDPC1    0.169 +/- 0.015
MORTGAGE30US0.168 +/- 0.016
day      0.165 +/- 0.014
GDP      0.161 +/- 0.016
Open     0.160 +/- 0.017
CP       0.157 +/- 0.016
CPIAUCSL0.156 +/- 0.015
month    0.156 +/- 0.015
UMCSENT 0.151 +/- 0.014
```

Figure 12: Feature importance from MLP classifier

Feature selection

- Merging important features from RandomForest and MLP as Feature Selection 1.
- Applying two filters to address multicollinearity and define the feature set as Feature Selection 2.
- Correlation threshold of 0.95 to identify and filter out highly collinear feature pairs, retaining only one feature from each pair
- Filter high VIF (Variance Inflation Factor) features which are high multicollinear with other features

Ultimately, we retain 12 features for df2 (Figure 13)

Ablation Testing

Ablation testing involves removing one feature at a time and observing the impact on model performance.

- Accuracy or f1_score drop (drop in positive value) means the removing of the feature degrades model performance
- Accuracy or f1_score gain (drop in negative value) means the removal of the feature upgrades model performance.

High Importance but Low Impact in Ablation:

- macd_signal, macd_hist, and macd are high importance features. Drop them should degrade model performance
- However, ablation tests show dropping macd oppositely increases accuracy. This may be due to macd being highly correlated with macd_signal we mentioned above (but not that high to be dropped in the feature selection stage), the model can rely on macd_signal when macd is dropped.
- For features highly correlated, the model can rely on remaining correlated features to make predictions, hence the low impact of their removal.

Features with High Impact in Ablation:

- PPIFIS and month, though not top important features, show significant drops in performance when removed, suggesting they provide unique information not captured by other features.

Negative Impact Features:

- Day, Open, Volume showing negative impact means the remove upgrades model performance despite their high importance. It may be due to the removal reducing overfitting or noise.

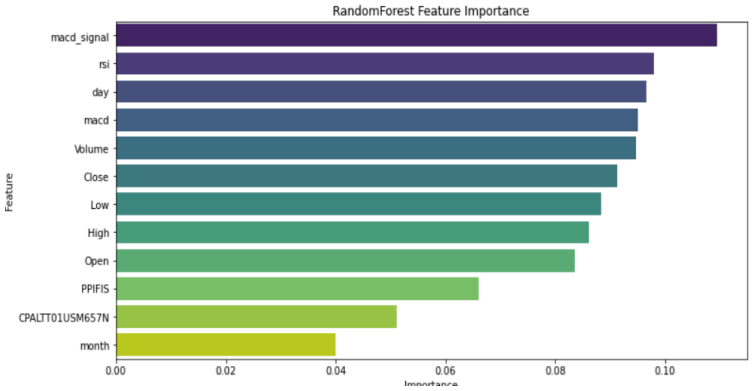


Figure 13: 12 most important features.

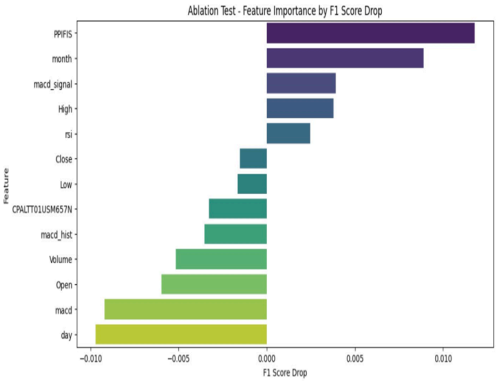


Figure 14: Feature importance by ablation test.

Sensitivity Analysis

Below is a feature sensitivity check on df1 by best model RandomForest. We scale each feature from 0.85 to 1.15 to see how model prediction performance on the test set drops.

Accuracy Drop

Feature	CGBD2534	CP	Close	DJIA	FEDFUNDS	GDP	GS10	HOUST	HSN1F	High	INDPRO	Low	NASDAQCOM	Open	PCE	PCU325211325211	PCU325325	RSXFS	SP500	UMCSENT	UNRATE	USACPALTT01CTGYM	Volume	WM2NS	ema	macd	macd_hist	macd_signal	rsi	sma
Scale Factor																														
0.85	−0.002	−0.005	0.011	0.000	0.005	0.000	0.000	0.000	−0.005	0.007	−0.007	0.009	0.007	0.000	−0.002	0.002	0.000	−0.002	0.007	0.005	0.009	0.005	0.009	0.011	−0.007	0.002	0.005	0.002	−0.007	0.005
0.90	−0.005	−0.002	0.009	−0.002	0.002	0.000	0.002	−0.002	−0.005	0.007	−0.007	0.009	0.002	0.002	0.000	0.005	−0.002	0.000	0.005	0.005	0.011	0.005	0.000	0.014	−0.007	−0.005	0.005	0.000	−0.005	0.009
0.95	0.000	0.000	0.007	−0.005	0.002	0.000	0.000	0.000	0.000	0.007	−0.002	0.011	0.007	0.005	0.000	0.002	0.000	−0.002	0.000	0.000	0.000	0.002	0.005	0.011	−0.011	−0.002	0.007	−0.011	0.002	0.002
1.00	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
1.05	0.000	0.000	0.005	0.016	0.005	0.000	−0.005	0.000	−0.002	0.002	0.005	0.005	−0.002	0.009	0.000	0.000	−0.002	−0.002	0.000	−0.002	0.000	0.000	−0.014	0.007	0.014	−0.007	−0.002	0.005	−0.009	0.005
1.10	0.002	0.002	0.011	0.009	0.005	0.000	−0.002	0.002	0.000	0.002	0.007	0.000	−0.005	0.007	0.005	0.005	−0.002	0.000	0.000	−0.007	0.002	0.000	−0.011	0.014	0.016	−0.002	0.000	0.000	−0.009	0.007
1.15	0.002	0.005	0.021	0.007	0.005	0.000	−0.005	0.002	0.000	0.000	0.009	0.002	−0.005	0.007	0.005	0.007	0.000	0.000	0.007	−0.005	0.005	0.000	−0.005	0.007	0.018	0.000	0.000	0.007	−0.009	0.009

Table 15: Sensitivity Analysis of Accuracy Loss

F1_score Drop

Feature	CGBD2534	CP	Close	DJIA	FEDFUNDS	GDP	GS10	HOUST	HSN1F	High	INDPRO	Low	NASDAQCOM	Open	PCE	PCU325211325211	PCU325325	RSXFS	SP500	UMCSENT	UNRATE	USACPALTT01CTGYM	Volume	WM2NS	ema	macd	macd_hist	macd_signal	rsi	sma
Scale Factor																														
0.85	−0.002	−0.005	0.012	−0.001	0.005	0.000	0.000	0.000	−0.005	0.005	−0.007	0.009	0.007	−0.000	−0.002	0.002	0.000	−0.003	0.007	0.004	0.010	0.005	0.007	0.011	−0.007	0.002	0.005	0.001	−0.006	0.003
0.90	−0.005	−0.002	0.010	−0.003	0.002	0.000	0.002	−0.002	−0.005	0.005	−0.007	0.009	0.003	0.002	0.000	0.005	−0.002	−0.000	0.005	0.005	0.012	0.005	−0.002	0.014	−0.007	−0.005	0.005	−0.001	−0.004	0.008
0.95	0.000	0.000	0.007	−0.005	0.002	0.000	0.000	0.000	−0.000	0.006	−0.002	0.012	0.008	0.005	0.000	0.002	0.000	−0.002	−0.000	0.000	−0.000	0.002	0.003	0.011	−0.012	−0.003	0.007	−0.012	0.002	0.001
1.00	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
1.05	0.000	0.000	0.005	0.016	0.005	0.000	−0.005	0.000	−0.002	0.002	0.005	0.004	−0.002	0.009	0.000	−0.000	−0.002	−0.002	0.000	−0.002	0.000	0.000	−0.013	0.006	0.013	−0.006	−0.003	0.005	−0.009	0.005
1.10	0.002	0.002	0.012	0.009	0.005	0.000	−0.003	0.002	−0.000	0.002	0.007	−0.001	−0.004	0.007	0.005	0.005	−0.002	0.000	−0.000	−0.006	0.002	0.000	−0.010	0.013	0.015	−0.002	−0.000	0.000	−0.009	0.007
1.15	0.002	0.005	0.022	0.007	0.005	0.000	−0.005	0.002	−0.000	0.000	0.009	0.002	−0.005	0.007	0.005	0.007	0.000	0.000	0.006	−0.004	0.004	0.000	−0.002	0.006	0.016	0.000	−0.000	0.007	−0.010	0.010

Table 16: Sensitivity Analysis of f1_score Loss

Our best model achieved an accuracy of 0.714 and an F1 score of 0.713. We set a 1% accuracy loss, approximately 0.007, as the threshold.
If the accuracy or F1 score loss falls between 0.007 and 0.014, we consider the model moderately sensitive to changes in that feature. If the loss exceeds 0.014, the model is deemed sensitive to the feature change. If the loss is less than 0.007, we consider the model insensitive to the feature change.
Thus we find the model is sensitive to Close, DJIA, ema, and moderately sensitive to Low, WM2NS, macd_signal, and rsi.

Identify important tradeoffs

Our evaluation metrics are accuracy, f1_score, confusion matrix on tw types

- Type A: Fit by train_test_split train set then predict by test set.
- Type B: Forward prediction.

We did not find any trade-offs among the evaluation metrics themselves. The main trade-off lies between the two types of performance evaluations (Type A and Type B).

- RandomForest achieves the best accuracy in type A, indicating its strong performance in fitting and predicting on the given dataset.
- However, RandomForest does not consistently outperform KNN and SVM in type B (forward prediction), particularly in short-day-fit then forward predictions.

Computational Expense vs. Practicality:

- RandomForest is computationally more expensive.
- KNN and SVM are less resource-intensive and perform comparably well in forward prediction, making them more suitable for real-time applications where quick predictions are necessary.

Failure analysis

Failure Analysis for Poor Forward Prediction of methodology #3 working on df2:

Though prediction in the test set shows good results by RandomForest, the forward prediction result is poor. Possible reasons are:

1. Data Leakage in train_test_split:
   In train_test_split data is randomly separated into training and testing sets. This random separation can lead to temporal data leakage, especially in time-series data. Such leakage allows the model to perform well in train_test_split but fail to generalize to forward predictions on new, unseen data.
2. Consecutive Patterns:
   Predicting 100 consecutive days in forward prediction introduces a challenge of capturing consecutive patterns. This issue is less pronounced in day-by-day predictions of Methodology #1 & #2, which shows better performance in forward prediction.
3. Training Data Relevance:
   Using a fixed 100 days as training data (Methodology #2, short-fit then predict) shows better performance than using increasing training lengths (Methodology #1, long-day fit then predict). This suggests that more recent data, even if fewer in number, is more relevant and beneficial for short-term predictions. Longer training periods with data far from the forward prediction days may degrade performance.

In summary, Methodology #3's underperformance highlights the pitfalls of data leakage and the need for models to adapt to consecutive patterns in temporal data.

Failure Analysis for Moderate Forward Prediction of Methodology #1 & #2 working on df1:

1. Though prediction in the test set shows good results (accuracy 0.714 in RandomForest) and best forward prediction shows moderate results (0.648 in KNN, with RandomForest and SVM similar level of prediction performance), we find a common problem that sometimes there is one day lag in prediction. It seems even best Methodology #2,

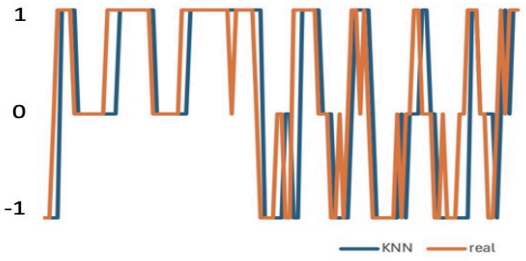


Fig 17: forward sometimes prediction shows one day lag of real target

short-day-fit then predict, still cannot timely capture the transition. This happens in all algorithms, not only KNN.

2. Data (Date) misalignment should be excluded. Since this methodology involves using the current day to predict movements 5 days later, it is unlikely that the observed lag is due to data misalignment between features and the target variable.
3. It's possible that using a larger window of data, such as incorporating data from 2-5 days, to predict the movement for a single day might improve the accuracy. This approach could help capture more relevant patterns and trends, providing better predictive power

Failure Analysis for Low Accuracy in Forward Prediction of Classes -1 and 1

Even our forward testing with methodology 2, the best performance methodology which uses the closest 100 days to predict the target day, revealed low accuracy for classes -1 (sell) and 1 (buy). Specifically, the accuracy for class -1 was around 40%-50%, and for class 1 it was 55%-60%. This is common in methodology #1 and #2.

- Class Distributions:
    - Original Distribution: 994:955:735 (1:0:-1)
    - Test Set Distribution: 842:701:641 (1:0:-1)
    - Forward Test Distribution: 152:254:94 (1:0:-1)

The primary reason for the poor performance in predicting classes -1 and 1 is the imbalanced distribution of the forward test set. In the forward test set, class 0 (hold) comprises more than 50% of the data, while class -1 is less than 20%, and class 1 is around 30%.

When using the closest 100 days for training, the model captures this imbalance. As a result, it becomes more proficient at predicting the majority class (0), but less effective for the minority classes (-1 and 1). This imbalance means the model has fewer examples of classes -1 and 1 to learn from, leading to lower accuracy in predicting these classes.

Unsupervised learning

Data sources

Share the same dataset of Supervised Learning

Feature Engineering

To transform the raw input data into features suitable for both supervised and unsupervised machine learning methods, the following steps were undertaken:

1. Extract Date and Close Price Columns

The initial step involved extracting the relevant columns from the raw dataset, specifically the 'Date' and 'Close' price columns. The 'Date' column was converted to a datetime format and set as the index of the DataFrame.

2. Resample to Daily Data

The time series data was resampled to daily frequency to ensure consistent temporal intervals. This was done by resampling the 'Close' price data using the mean value for each day.

3. Handle Missing Values

Missing values in the time series data were handled using forward fill, which propagates the last known value forward to fill any gaps.

4. Segment the Time Series

The time series data was segmented into smaller windows for analysis. The window size defined the length of each segment, and the shifting size defined the step size for sliding the window across the data.

5. Normalize the Time Series Data

The segments were normalized to ensure uniformity in scale, with each segment scaled to have zero mean and unit variance.

Methods description

We implemented an unsupervised learning approach to analyze normalized time series segments and uncover underlying patterns. Our workflow included two distinct clustering methods: K-Shape clustering and DTW K-Means clustering. These methods were chosen due to their unique mechanisms and suitability for time series data.

K-Shape Clustering

K-Shape clustering, an extension of the traditional K-Means algorithm for time series data, was applied to identify patterns in the time series segments. This method was selected because it addresses specific challenges in time series clustering through its unique features:

Shape-Based Distance Measure:

K-Shape utilizes a shape-based distance measure, which focuses on the overall shape of the time series rather than individual pointwise differences. This approach captures patterns in time series data more effectively, making it well-suited for the BTC price movement analysis.

The shape-based distance between two time series are calculated as follow:

$$D_{\text{shape}}(\mathbf{T}_i, \mathbf{T}_j) = 1 - \max_\tau \left(\frac{\text{CC}_{\mathbf{T}_i,\mathbf{T}_j}(\tau)}{\sqrt{\sum_{t=1}^{n} T_i^2(t) \cdot \sum_{t=1}^{n} T_j^2(t+\tau)}} \right)$$

The shape-based distance metric used in K-Shape clustering is specifically designed to capture the overall shape and pattern of time series data. Unlike traditional distance metrics that focus on pointwise differences, the shape-based distance leverages normalized cross-correlation to align and compare time series, ensuring that the most similar shapes are grouped together even if they are temporally misaligned.

DTW K-Means Clustering

We also employed DTW K-Means clustering, which uses the Dynamic Time Warping (DTW) distance metric. This method was chosen due to its effectiveness in handling time series data with varying speeds or temporal distortions. The DTW algorithm stretches and compresses segments of the series to achieve the best match between the sequences. This non-linear alignment allows DTW to handle time series with different lengths and temporal distortions effectively.

DTW is an algorithm for measuring similarity between two temporal sequences that may vary in time or speed. Unlike traditional distance metrics, such as Euclidean distance, DTW finds an optimal alignment between time series.

Dynamic Time Warping (DTW):

DTW measures the similarity between time series by finding the optimal alignment between them, allowing for non-linear distortions. This makes DTW K-Means particularly effective for datasets where patterns may occur at different speeds or with temporal variations.

Principal Component Analysis (PCA)

Principal Component Analysis (PCA) was applied to the preprocessed data to reduce its dimensionality while retaining as much variance as possible. This method was selected due to its ability to transform high-dimensional data into a lower-dimensional form, making it easier to analyze and interpret.

Explained Variance Ratio:

PCA was used to determine the proportion of the dataset's variance captured by each principal component (PC). The goal was to retain at least 95% of the variance, ensuring that the reduced dimensions still represented the original data effectively.

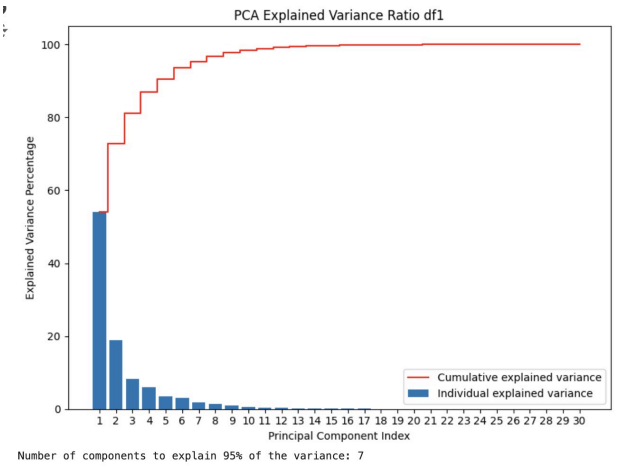


Figure 18: PCA Explained Variance Ratio for BTC Time Series Data

Most Important Features:

For each principal component, the feature with the highest absolute loading was identified. This feature was considered the most important for that principal component, indicating its significant contribution to the variance captured by the PC.

Combined Importance:

The combined importance metric was calculated as the product of the explained variance ratio and the feature's absolute loading. This metric provides a holistic measure of each feature's contribution to the overall variance captured by the PCA model.

To visualize the importance of each PCA component, we created a bar plot that highlights both the variance ratio and the combined importance of each principal component. The blue bars represent the variance ratio for each principal component. This metric indicates the proportion of the total variance in the dataset that each component captures. The red bars represent the combined importance, calculated as the product of the variance ratio and the feature importance. For each principal component, we identified the feature with the highest absolute loading, referred to as the most important feature. These features are crucial as they heavily influence the direction of the component.

	PC	Most_Important_Feature	Variance_Ratio	Importance	Combined_Importance
0	PC1	SP500	0.540	0.243	0.131
1	PC2	GS10	0.188	0.377	0.071
2	PC3	macd	0.082	0.502	0.041
3	PC4	macd_hist	0.060	0.458	0.027
4	PC5	macd_hist	0.036	0.578	0.021
5	PC6	HSN1F	0.030	0.555	0.017
6	PC7	Volume	0.017	0.779	0.014

Table 19: Important Features for Principal Components in BTC Time Series Data

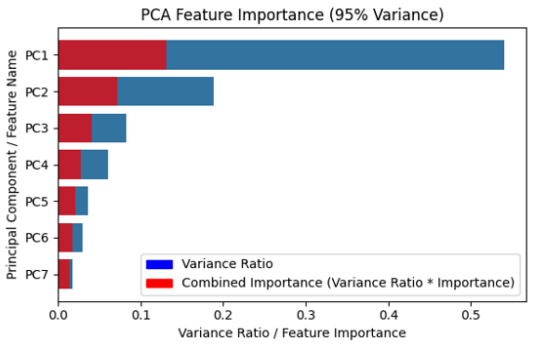


Figure 20 : PCA Feature Importance for BTC Time Series Data (95% Variance)

The PCA feature importance plot reveals that the first two principal components (PC1 and PC2) are the most significant in capturing the variance in the dataset. Subsequent principal components contribute less to the explained variance, indicating that a few PCs can effectively summarize the data. The most important features identified for each PC are crucial for understanding the underlying patterns and structure of the dataset. This analysis emphasizes the utility of PCA in dimensionality reduction and feature extraction, facilitating better interpretation and further analysis of the data.

Optimal Number of Clusters:

The optimal number of clusters was determined using the elbow plot method. By plotting the within-cluster sum of squares

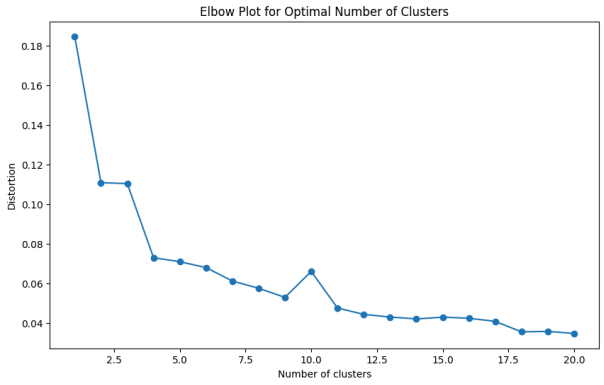


Figure 21: Elbow Plot for Determining the Optimal Number of Clusters

against the number of clusters, we identified the point where the rate of decrease slows down.

As we can observe in the figure, the optimal number of clusters is 5.

Time series decomposition

Time series decomposition involves separating a time series into several distinct components:

- Trend: The long-term movement or direction in the data.
- Seasonality: The repeating short-term cycle or pattern within the data.
- Residuals: The remaining part after removing the trend and seasonality, representing noise or random fluctuations.

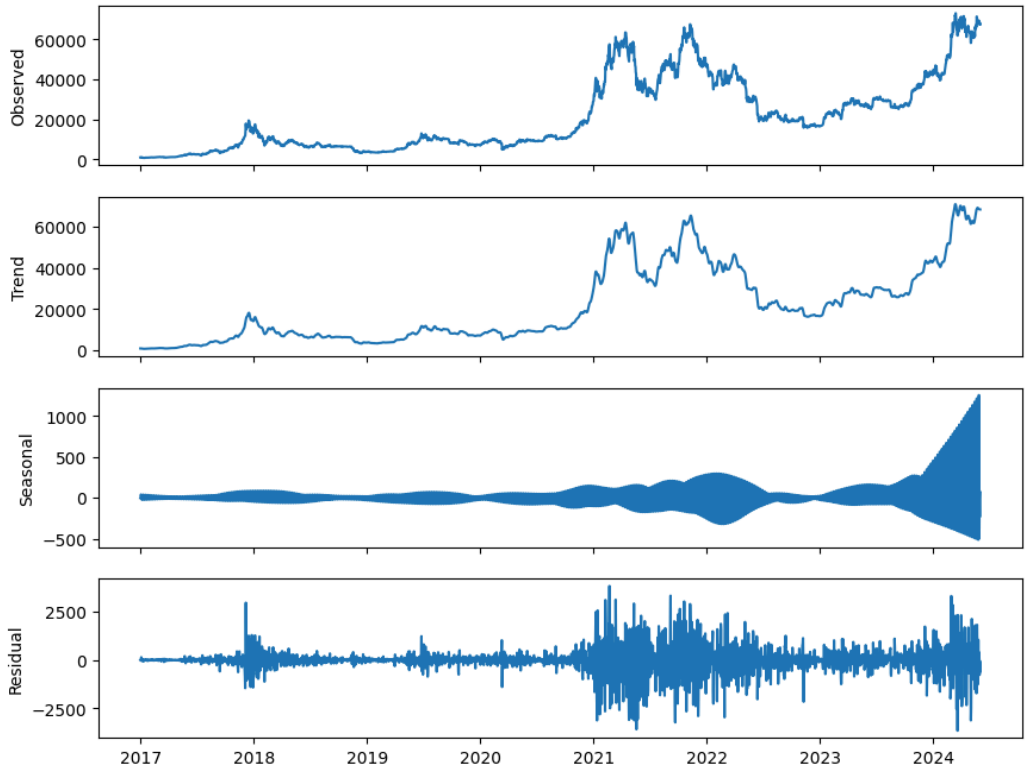


Figure 22: Seasonal Decomposition of BTC Time Series Data

The seasonal decomposition plot effectively breaks down the BTC time series data into understandable components. The observed data shows significant fluctuations and an overall upward trend. The trend component highlights the long-term growth of BTC prices, while the seasonal component reveals repeating patterns with increasing amplitude. The residual component captures the unexplained variability, showing higher volatility during significant market events.

Unsupervised evaluation

Overall results

Evaluation metrics

The clusters were evaluated using Silhouette Score, Davies-Bouldin Index, and inertia to measure the quality of the clustering.

Silhouette Score

The Silhouette Score measures how similar an object is to its own cluster compared to other clusters. It provides a combined measure of cohesion and separation.

Davies-Bouldin Index

The Davies-Bouldin Index is a metric for evaluating clustering algorithms based on the average similarity ratio of each cluster with its most similar cluster. Lower values indicate better clustering.

Inertia

Inertia measures the sum of squared distances between each point and the centroid of its assigned cluster. It is a measure of how internally coherent the clusters are.

To inspect each cluster's overall shape and pattern, we generated plots for each cluster showing the individual time series and their respective cluster centers.

Result

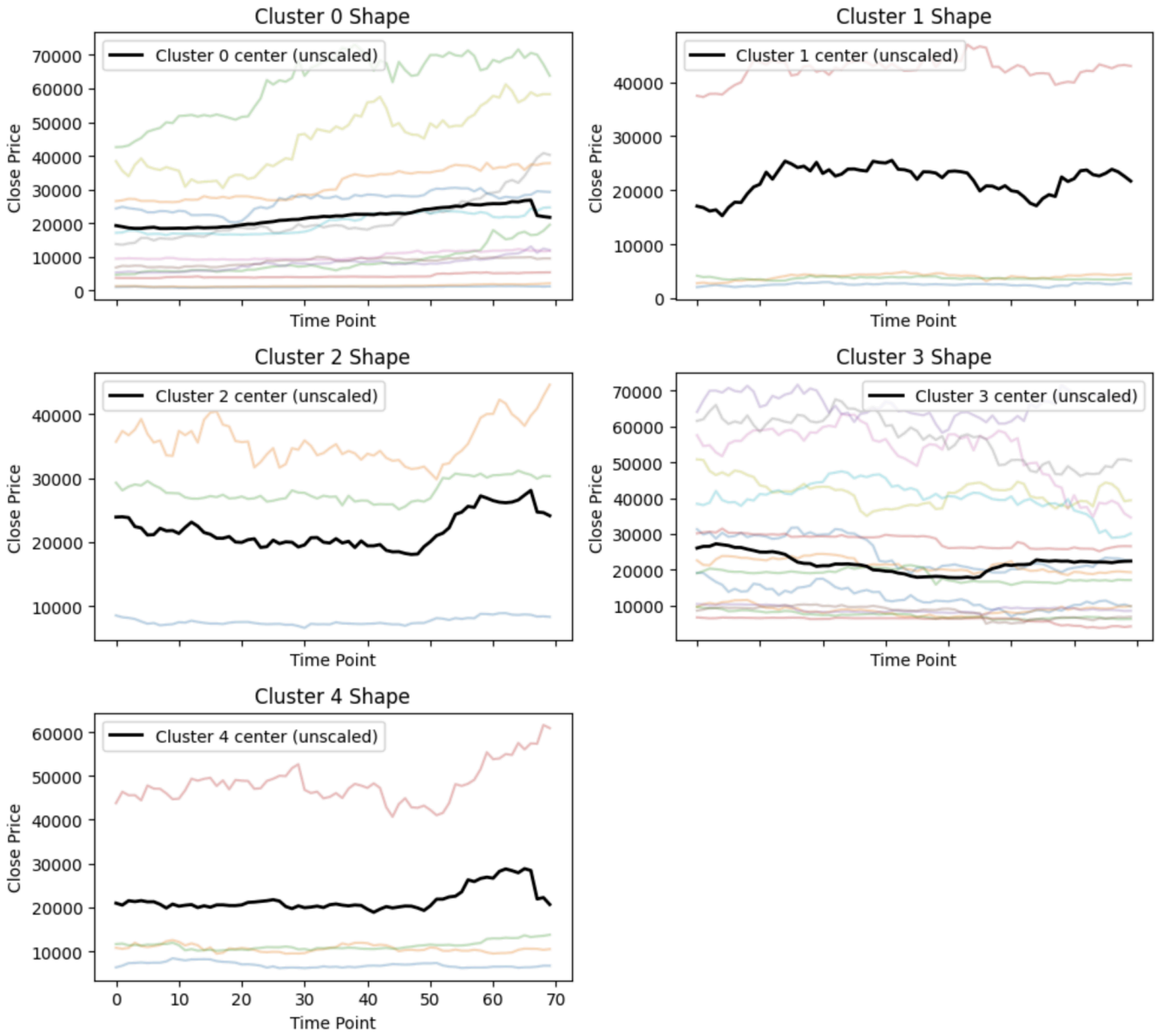


Figure 23: Shape Patterns of BTC Time Series Clusters Using K-Shape Clustering

Observation

The clusters exhibit distinct patterns in the time series data. Cluster 0 shows moderate fluctuations with a slight upward trend, indicating gradual growth. Cluster 1 is relatively stable with a flat trend. Cluster 2 features varied trends with a noticeable spike towards the end, suggesting sudden growth events. Cluster 3 displays high volatility and a general downward trend, indicating periods of decline. Cluster 4 shows a mixture of stable and slightly increasing trends. Overall, the clustering effectively segmented the data into meaningful groups, each with unique trends and behaviors.

Sensitivity analysis

With 5 as the optimal number of clusters, we iterated through different window sizes, ran the clustering algorithm and calculated the evaluation metrics, as displayed below.

window_size	30.000000	40.000000	50.000000	60.000000	70.000000	80.000000
n_clusters	5.000000	5.000000	5.000000	5.000000	5.000000	5.000000
silhouette_score	0.145350	0.064467	0.081408	0.179084	0.199880	0.030742
davies_bouldin_index	3.133176	2.433239	2.528281	1.705322	1.668406	2.693607
inertia	408.000000	361.000000	460.000000	265.000000	145.000000	239.000000

Table 24. Sensitivity Analysis for Window Size Change

The sensitivity analysis reveals that the K-Shape model's performance improves with larger window sizes. As the window size increases, the silhouette score and Davies Bouldin Index both indicate better-defined and more distinct clusters, while inertia decreases, showing more compact clusters. Therefore, the clustering results are sensitive to the choice of window size, with larger windows providing more robust and meaningful clustering outcomes. The optimal window size for K-Shape clustering, based on this evaluation, appears to be around 70, as it provides the best silhouette score, lowest Davies-Bouldin index, and lowest inertia. This indicates that clusters are well-defined, distinct, and compact at this window size.

Discussion

Supervised lessons

Limitations of Train-Test Split:

The traditional train_test_split approach, which randomly assigns data points to the training and testing sets, cannot predict as accurately in forward testing scenarios. This limitation is not only due to data imbalance but also because of the consecutive nature of time-series data. In methodology #3, where we used a long-day fit and predicted group by group with each group representing 100 consecutive days, the model struggled to generalize well to consecutive future data points.

Challenges of Forward Testing:

Even when employing methodology #2, which uses short-day fits to predict day by day, forward testing performance was still worse than baseline. This indicates that data imbalance significantly impacts model accuracy. The forward testing accuracy drops further highlight the difficulty in predicting highly volatile and imbalanced time-series data, such as BTC movements, even with shorter, more recent training windows.

What Surprise Us:

Even day by day forward prediction cannot get the same level of accuracy as the test set.
RandomForest doesn’t outperform KNN in forward testing

How could we extend our work:

Learn from Unsupervised work to see if it assists supervised model prediction. Add NLP analysis on twitter and news contents to see whether it is helpful on model prediction. Go deep on outliers analysis to understand why multiple days' close price in a row is treated differently and it may reflect some important information which we missed but is crucial to model prediction.

Unsupervised lessons

Future Extensions:

Future work could extend the unsupervised learning approach by integrating additional data sources, such as sentiment analysis from social media and news articles, to further enrich the clustering analysis. Additionally, incorporating advanced outlier detection methods could help in understanding and mitigating the impact of extreme price movements on the clustering results.

Unexpected Flatness of Cluster Centroids:

A surprising observation was the distinct cluster centroids of the K-Shape clustering, which were notably flat. This was unexpected given the high volatility typically associated with BTC price movements, suggesting that the clustering method effectively smoothed out short-term fluctuations to capture underlying trends.

Ethical considerations

Both supervised learning and unsupervised learning used public data of Bitcoin (BTC) time series and FRED Indices, there are no major ethical issues anticipated. The data used is publicly available, and the clustering process itself does not inherently introduce significant ethical concerns. However, it is still important to use the insights responsibly and ensure compliance with relevant financial regulations to prevent any unintended consequences. We make sure that we present data and findings honestly without misrepresentation or manipulation to support a particular agenda.

Statement of work

Irene YiJu Su	Khanh Nguyen	Xinmei Luan
leading the project, find out data source and get the data ready, df0 and df1 all analysis, methodology #1 and 2,PCA, sensitive anlysis, failure analysis,write and combine the report.	unsupervised learning: feature engineering, comparison on K-shaped clustering and DTW K-means clustering,chose optimal number of clusters, time series decomposition, writing report.	supervised learning: work on df2,methodoloy #3,outliers recognition, multicollinearity analyis, feature selection, check the difference between train_test_split and forward testing, testing different method on forecast accuracy improvement.

We use chatGPT to refine our wordings.

Appendix A - Bibliography

[1] J Rajasombat. (2021,June 17). “Using Supervised Machine Learning Models to Predict Equity Price Movements”. Linkedin ,https://www.linkedin.com/pulse/using-supervised-machine-learning-models-predict-price-rajasombat/. Accessed June 2024.

[2]R Gupta, M Chen.”Sentiment Analysis for Stock Price Prediction”.2020 IEEE Conference on Multimedia Information Processing and Retrieval (MIPR), 2020, pp. 213-218. https://ieeexplore.ieee.org/document/9175549

[3]S Mohan, S Mullapudi. “Stock Price Prediction Using News Sentiment Analysis”,2019 IEEE Fifth International Conference on Big Data Computing Service and Applications (BigDataService),2019,pp.205-208. https://ieeexplore.ieee.org/document/8848203

[4]N Darapaneni, A Paduri, H Sharma,”Stock Price Prediction using Sentiment Analysis and Deep Learning for Indian Markets “.Quantitative Finance.Apr,7 2022.pp.1-15.https://arxiv.org/abs/2204.05783

Appendix B - All Features

Number	Feature	Feature Description	Dtype
0	Date	Date	object
1	Open	Open	float64
2	High	High	float64
3	Low	Low	float64
4	Close	Close	float64
5	Volume	Volume	int64
6	sma	Simple Moving Average	float64
7	ema	Exponential Moving Average	float64
8	macd	Moving Average Convergence Divergence	float64
9	macd_signal	macd_signal	float64
10	macd_hist	macd_hist	float64
11	rsi	Relative Strength Index	float64
12	nextClose	next 5 days' close	float64
13	target	target	int64
14	SP500	S&P 500 Index	float64
15	NASDAQCOM	NASDAQ Composite Index	float64
16	DJIA	Dow Jones Industrial Average	float64
17	CP	Consumer Price Index for All Urban Consumers	float64
18	RSXFS	Retail Sales Excluding Food Services	float64
19	WM2NS	M2 Money Stock	float64
20	GDP	Gross Domestic Product	float64
21	PCU325211325211	Producer Price Index by Industry: Plastics Material and Resin Manufacturing	float64
22	PCE	Personal Consumption Expenditures	float64
23	USACPALTT01CTGYM	Harmonized Consumer Price Index for the United States	float64
24	UNRATE	Civilian Unemployment Rate	float64
25	CGBD2534	Civilian Employment-Population Ratio: 25 to 34 years	float64
26	FEDFUNDS	Effective Federal Funds Rate	float64
27	GS10	10-Year Treasury Constant Maturity Rate	float64
28	UMCSENT	University of Michigan: Consumer Sentiment	float64
29	INDPRO	Industrial Production Index	float64
30	HSN1F	New One Family Houses Sold	float64
31	HOUST	Housing Starts: Total New Privately Owned Housing Units Started	float64
32	PCU325325	Producer Price Index by Industry: Agricultural Chemical Manufacturing	float64
33	CPIAUCSL	Consumer Price Index for All Urban Consumers: All Items	float64
34	MEDCPIM158SFRBCLE	Median Consumer Price Index	float64
35	CPIAUCSL	Consumer Price Index for All Urban Consumers: All Items in U.S.	float64
36	MEDCPIM158SFRBCLE	Graph and download economic data for Median Consumer Price Index from Jan 1983 to May 2024 about CPI, median, rate, price index, indexes, price, and USA.	float64
37	CORESTICKM159SFRBAT	16% Trimmed-Mean Consumer Price Index	float64
38	CPALTT01USM657N	Consumer Price Index: Total All Items for the United States	float64
39	STICKCPIM157SFRBATL	Sticky Price Consumer Price Index	float64
40	PPIACO	Producer Price Index by Commodity for All Commodities	float64
41	PCUOMFGOMFG	Producer Price Index by Industry: Total Manufacturing Industries	float64
42	PCU325211325211	Producer Price Index by Industry: Plastics Material and Resin Manufacturing	float64
43	PCU325325	Producer Price Index by Industry: Chemical Manufacturing	float64
44	PPIFIS	Producer Price Index by Industry: Financial Services	float64
45	DFF	Effective Federal Funds Rate	float64
46	REAINTRATREARAT10Y	Real Interest Rate, 10-Year	float64
47	MORTGAGE30US	30-Year Fixed Rate Mortgage Average	float64
48	A091RC1Q027SBEA	Real Gross Domestic Product: Annual Percentage Change	float64
49	DGS2	2-Year Treasury Constant Maturity Rate	float64
50	GDPC1	Real Gross Domestic Product	float64
51	T10YIE	10-Year Breakeven Inflation Rate	float64
52	WM2NS	Seasonally adjusted M2	float64
53	M2REAL	Real M2 Money Stock	float64
54	UNRATE	Civilian Unemployment Rate	float64

Appendix C

df0 fitting result with confusion matrix

model	ConfusionMatrix	Accuracy	f1_score	recall	precision
RandomForest	[[87, 29, 8], [23, 85, 34], [12, 30, 129]]	0.688787	0.689046	0.688787	0.689349
KNN	[[76, 33, 15], [28, 77, 37], [19, 40, 112]]	0.606407	0.607639	0.606407	0.609363
DecisionTree	[[78, 27, 19], [35, 78, 29], [28, 33, 110]]	0.608696	0.609741	0.608696	0.613060
SVM	[[66, 32, 26], [27, 77, 38], [27, 48, 96]]	0.546911	0.547848	0.546911	0.550214
NaiveBayes	[[38, 56, 30], [39, 70, 33], [37, 65, 69]]	0.405034	0.405441	0.405034	0.418219
LogisticRegression	[[17, 40, 67], [15, 59, 68], [21, 38, 112]]	0.430206	0.401631	0.430206	0.408387

df1 fitting result with confusion matrix

model	ConfusionMatrix	Accuracy	f1_score	recall	precision
RandomForest	[[99, 20, 12], [24, 81, 30], [9, 30, 132]]	0.713959	0.713258	0.713959	0.712694
KNN	[[98, 22, 11], [26, 80, 29], [18, 27, 126]]	0.695652	0.695055	0.695652	0.695480
DecisionTree	[[87, 31, 13], [41, 62, 32], [16, 40, 115]]	0.604119	0.604512	0.604119	0.606372
SVM	[[93, 25, 13], [30, 78, 27], [12, 38, 121]]	0.668192	0.669452	0.668192	0.671489
NaiveBayes	[[46, 6, 79], [30, 21, 84], [37, 10, 124]]	0.437071	0.400349	0.437071	0.466432
LogisticRegression	[[52, 29, 50], [39, 48, 48], [38, 31, 102]]	0.462243	0.457117	0.462243	0.457703